Assignment -4

Python Programming

Assignment Date	6 November 2022
Student Name	Bhuvana R
Student Roll Number	311419205006
Maximum Marks	2 Marks

Description:- Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from google.colab import drive
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
```

DATASET LOADED

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mou
```

```
path='/content/drive/MyDrive/Colab Notebooks/IBM Project/abalone.csv'
```

+ Code + Text

```
df=pd.read_csv(path)
```

```
df.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
df.tail()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age

```
df.describe()
```

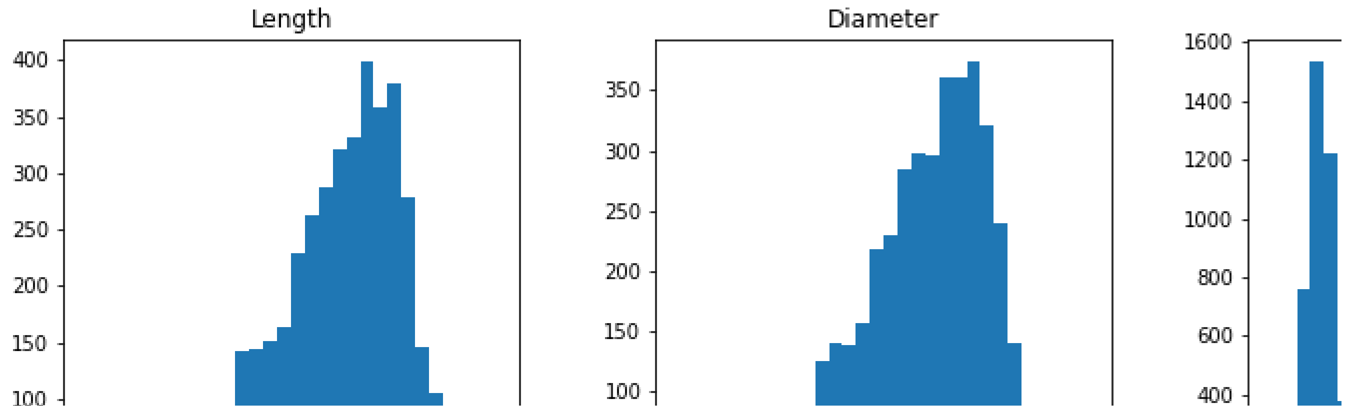
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.

```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

Univariate Analysis

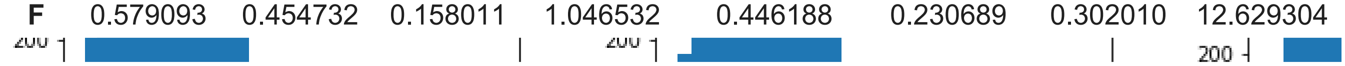
```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f50c63ffc90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f50c77a4e50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f50c6311390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f50c62c8990>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f50c627ff90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f50c62405d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f50c61f8c50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f50c61be1d0>]],
      dtype=object)
```




```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

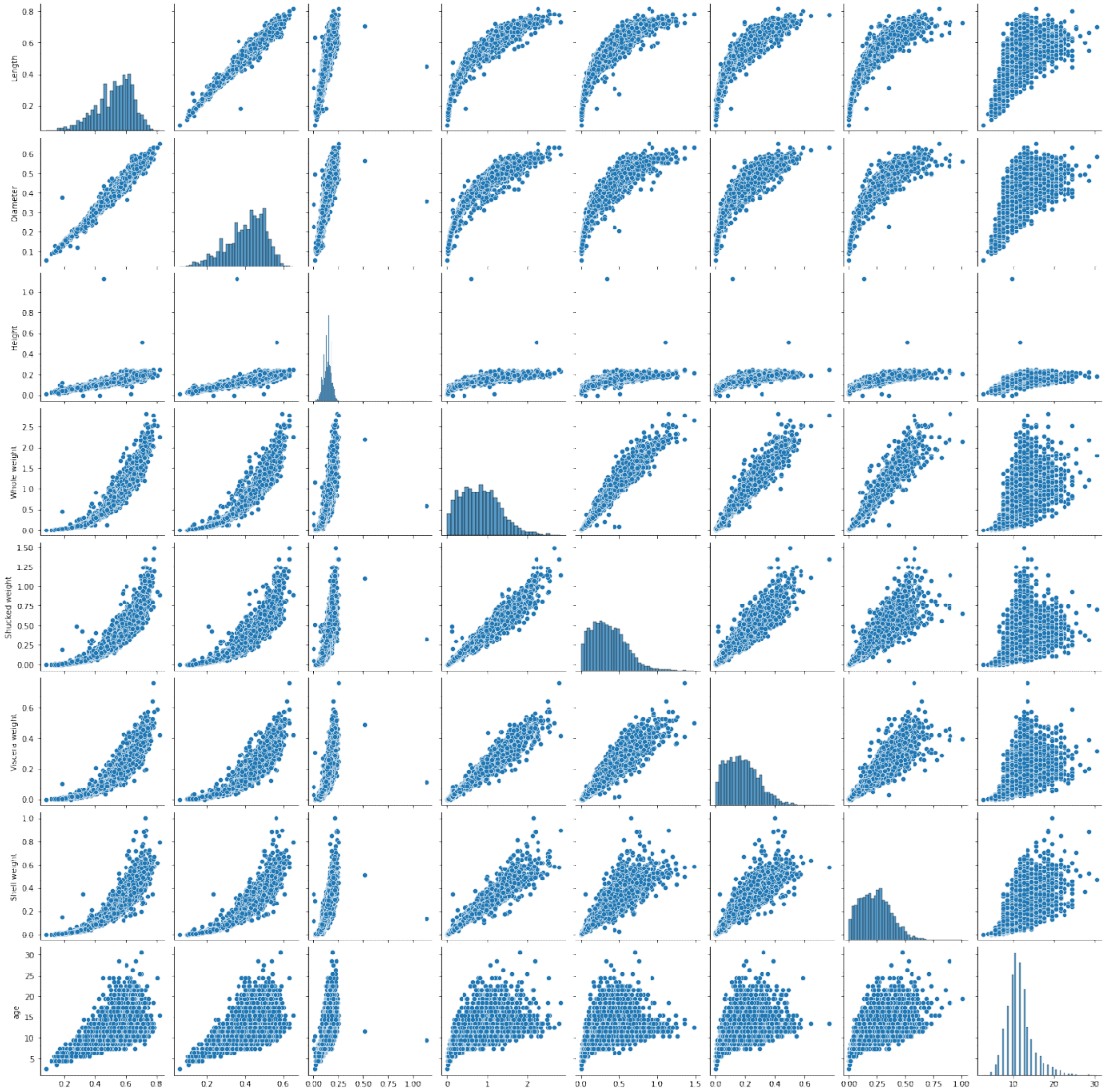
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
Sex								
I	0.427746	0.326494	0.107996	0.431363	0.191035	0.092010	0.128182	9.390462
M	0.561391	0.439287	0.151381	0.991459	0.432946	0.215545	0.281969	12.205497
F	0.579093	0.454732	0.158011	1.046532	0.446188	0.230689	0.302010	12.629304



Bivariate and Multivariate Analysis


```
numerical_features = df.select_dtypes(include = [np.number]).columns
sns.pairplot(df[numerical_features])
```

<seaborn.axisgrid.PairGrid at 0x7f50c5cfee50>



Descriptive Statistics

```
df.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.

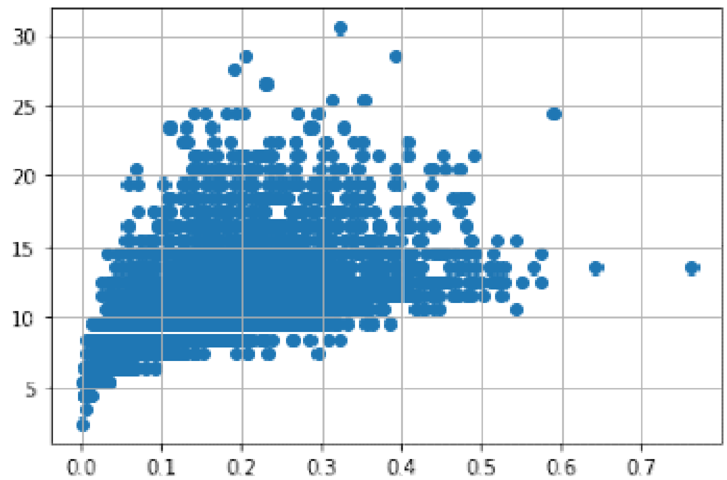
Check for missing values

```
df.isnull().sum()
```

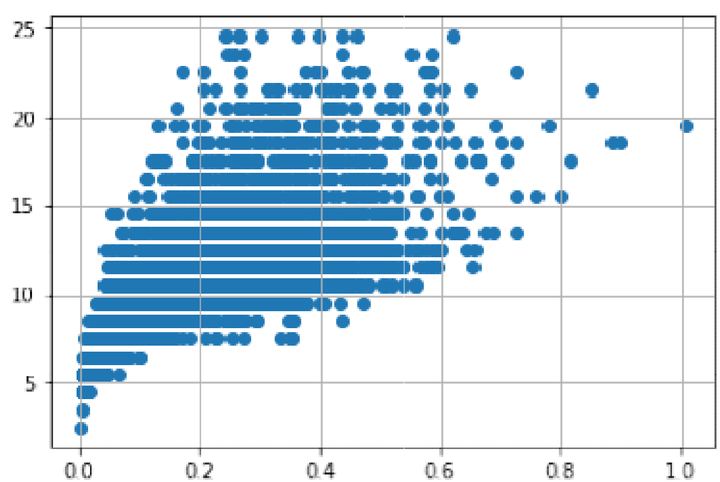
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

Outlier Handling

```
df = pd.get_dummies(df)
dummy_data = df.copy()


#outliers removal for viscera weight


var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

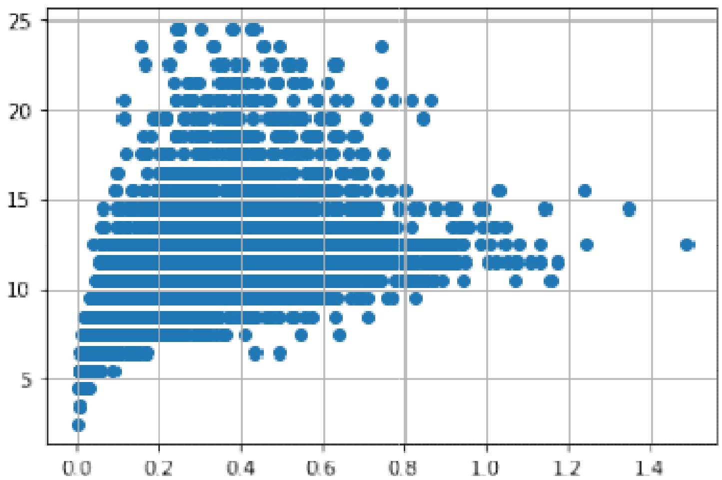



```
#outliers removal for shell weight


var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
df.drop(df[(df['Shell weight']> 0.6) & (df['age'] < 25)].index, inplace=True)
df.drop(df[(df['Shell weight']<0.8) & (df['age'] > 25)].index, inplace=True)
```



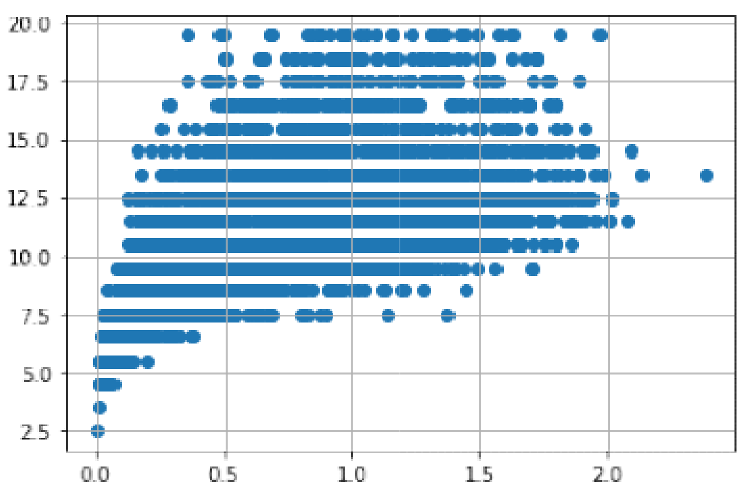
```
#Outliers removal for shuked weight
```

```
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
df.drop(df[(df['Shucked weight']>= 1) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Shucked weight']<1) & (df['age'] > 20)].index, inplace=True)
```




```
#outliers removal for whole weight
```

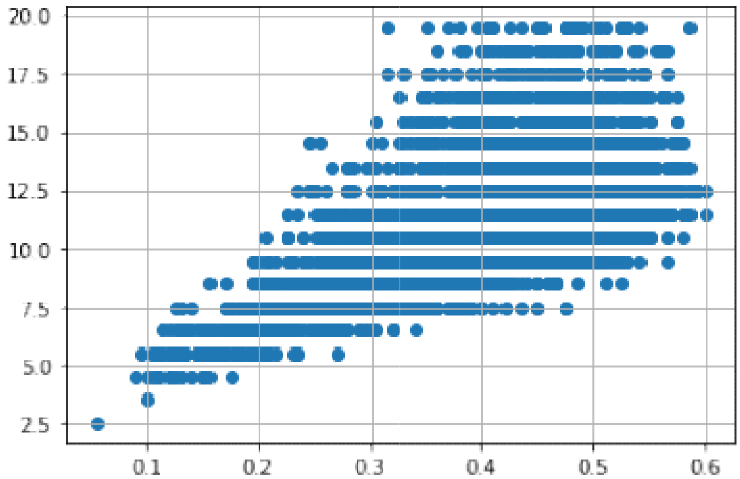
```
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
df.drop(df[(df['Whole weight'] >= 2.5) &(df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight']<2.5) & (df['age'] > 25)].index, inplace = True)
```



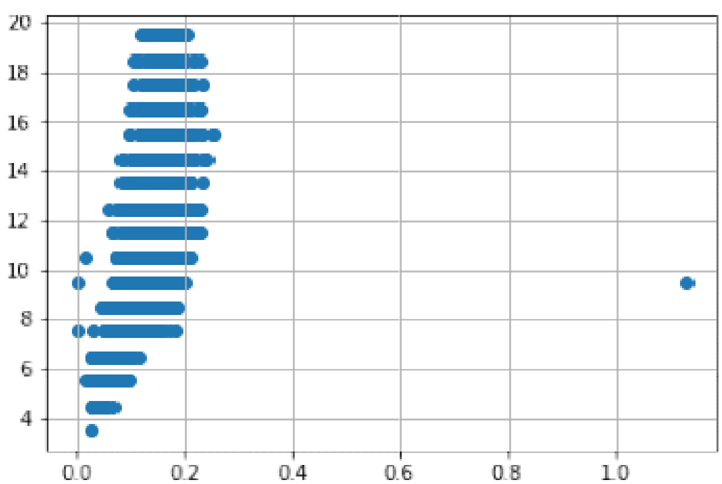

```
#outliers removal for diameters
```

```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
```

```
plt.grid(True)
df.drop(df[(df['Diameter'] <0.1) &(df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Diameter']<0.6) & (df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Diameter']>=0.6) & (df['age'] < 25)].index, inplace = True)
```

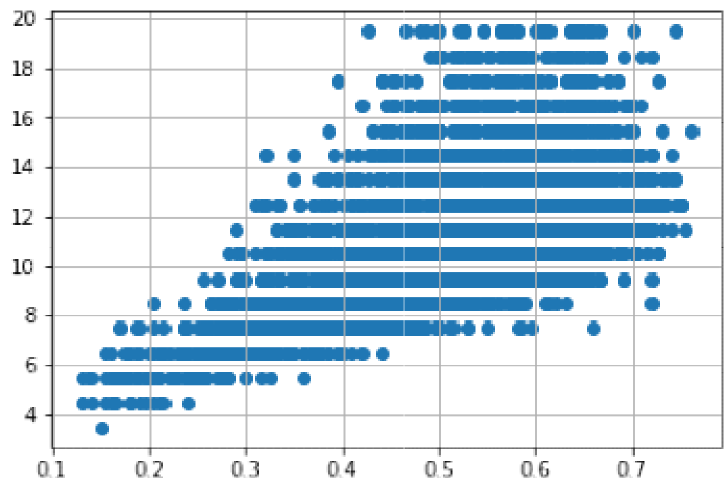



```
#outliers removal for height


var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
df.drop(df[(df['Height'] > 0.4) &(df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (df['age'] > 25)].index, inplace = True)
```




```
#outliers removal for length


var = 'Length'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
df.drop(df[(df['Length'] <0.1) &(df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Length']<0.8) & (df['age'] > 25)].index, inplace = True)
```

```
df.drop(df[(df['Length']>=0.8) & (df['age'] < 25)].index, inplace = True)
```



Categorical Columns

```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `np
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/rele
```

```
numerical_features
```

```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
      dtype='object')
```

```
categorical_features
```

```
Index([], dtype='object')
```

Split the dependent and independent variables

```
x=df.iloc[:,:5]
y=df.iloc[:,5:]
```

```
x
```

	Length	Diameter	Height	Whole weight	Shucked weight
0	0.455	0.365	0.095	0.5140	0.2245
1	0.350	0.265	0.090	0.2255	0.0995
2	0.530	0.420	0.135	0.6770	0.2565
3	0.440	0.365	0.125	0.5160	0.2155
4	0.330	0.255	0.080	0.2050	0.0895
...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700
4173	0.590	0.440	0.135	0.9660	0.4390
4174	0.600	0.475	0.205	1.1760	0.5255
4175	0.625	0.485	0.150	1.0945	0.5310

y

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
0	0.1010	0.1500	16.5	0	0	1
1	0.0485	0.0700	8.5	0	0	1
2	0.1415	0.2100	10.5	1	0	0
3	0.1140	0.1550	11.5	0	0	1
4	0.0395	0.0550	8.5	0	1	0
...	...	...	...	...	...	...
4172	0.2390	0.2490	12.5	1	0	0
4173	0.2145	0.2605	11.5	0	0	1
4174	0.2875	0.3080	10.5	0	0	1
4175	0.2610	0.2960	11.5	1	0	0
4176	0.3765	0.4950	13.5	0	0	1

3995 rows × 6 columns

split the data (train and test)

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Model Building

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

```
LinearRegression()
```

Train the model

```
x_train[0:4]
```

	Length	Diameter	Height	Whole weight	Shucked weight
2423	0.410	0.315	0.110	0.3210	0.1255
1216	0.310	0.225	0.070	0.1055	0.4350
3002	0.645	0.505	0.185	1.4630	0.5920
985	0.570	0.450	0.155	1.1935	0.5130

```
y_train[0:5]
```

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
2423	0.0655	0.0950	11.5	1	0	0
1216	0.0150	0.0400	6.5	0	1	0
3002	0.3905	0.4160	11.5	0	0	1
985	0.2100	0.3430	11.5	0	0	1
2838	0.2330	0.2595	10.5	0	0	1

```
x_test[0:4]
```

	Length	Diameter	Height	Whole weight	Shucked weight
3006	0.700	0.545	0.185	1.6135	0.750
3817	0.475	0.385	0.120	0.5620	0.289
4094	0.630	0.530	0.175	1.4135	0.667
402	0.435	0.325	0.110	0.4335	0.178

```
y_test[0:5]
```

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
3006	0.4035	0.3685	12.5	0	0	1
3817	0.0905	0.1530	9.5	0	0	1
4094	0.2945	0.3555	14.5	0	0	1
402	0.0985	0.1550	8.5	1	0	0

```
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
```

```
lrpred=lr.predict(x_test[0:9])
```

```
lrpred
```

```
array([[ 0.35064154,  0.42317517, 12.55339604,  0.50780283, -0.08545215,
         0.57764932],
       [ 0.11701718,  0.15625023,  9.84878154,  0.23508899,  0.45415266,
         0.31075835],
       [ 0.30007654,  0.37892926, 12.30238534,  0.50574715, -0.05317174,
         0.54742459],
       [ 0.09692013,  0.13181165,  9.95964476,  0.18232777,  0.5578356 ,
         0.25983664],
       [ 0.25590426,  0.32122087, 11.92694455,  0.41939293,  0.12392858,
         0.45667849],
       [ 0.15846252,  0.20923024, 11.29126176,  0.29014005,  0.36997235,
         0.33988761],
       [ 0.28730637,  0.35538064, 12.37098073,  0.43130339,  0.09697514,
         0.47172147],
       [ 0.15229535,  0.20263728, 10.84591436,  0.29722028,  0.34107547,
         0.36170425],
       [ 0.05210596,  0.07789379,  9.1755676 ,  0.12539739,  0.65136117,
         0.22324144]])
```

Measure the performance using Metrics

```
r2_score(lr.predict(x_test),y_test)
```

```
-3.1758408437233587
```